TVMCon 2021

VTA++: Expanded Design Space Exploration with an Enhanced Versatile Tensor Accelerator

Suvadeep Banerjee, Steve Burns, Pasquale Cocchini, Abhijit Davare, Shweta Jain, Desmond Kirkpatrick, Anton Sorokin, Jin Yang, Zhenkun Yang

Intel Labs

intel®

Motivation: Design Methodology Research

- Inspired by 2018 Turing Award paper [1]:
  - Domain-specific languages (DSLs) and architectures (DSAs) are key.
  - Gains "... will require a vertically integrated design team that understands applications, domain-specific languages and related compiler technology, computer architecture and organization, and the underlying implementation technology."
- Research Goal
  - Lower the design cost for DSLs deployed onto DSAs
- Hypothesis
  - Incremental feature addition with a vertical development stack
  - Neither software-first nor hardware-first design
- TVM/VTA was an appealing starting point
  - Development stack spans workload down to hardware, yet fast simulation possible
  - User-schedulable compiler provides rich software choices
  - Parameterized hardware presents a large design space

[1] Hennessy, John L., and David A. Patterson. "A new golden age for computer architecture." *Communications of the ACM* 62.2 (2019): 48-60.

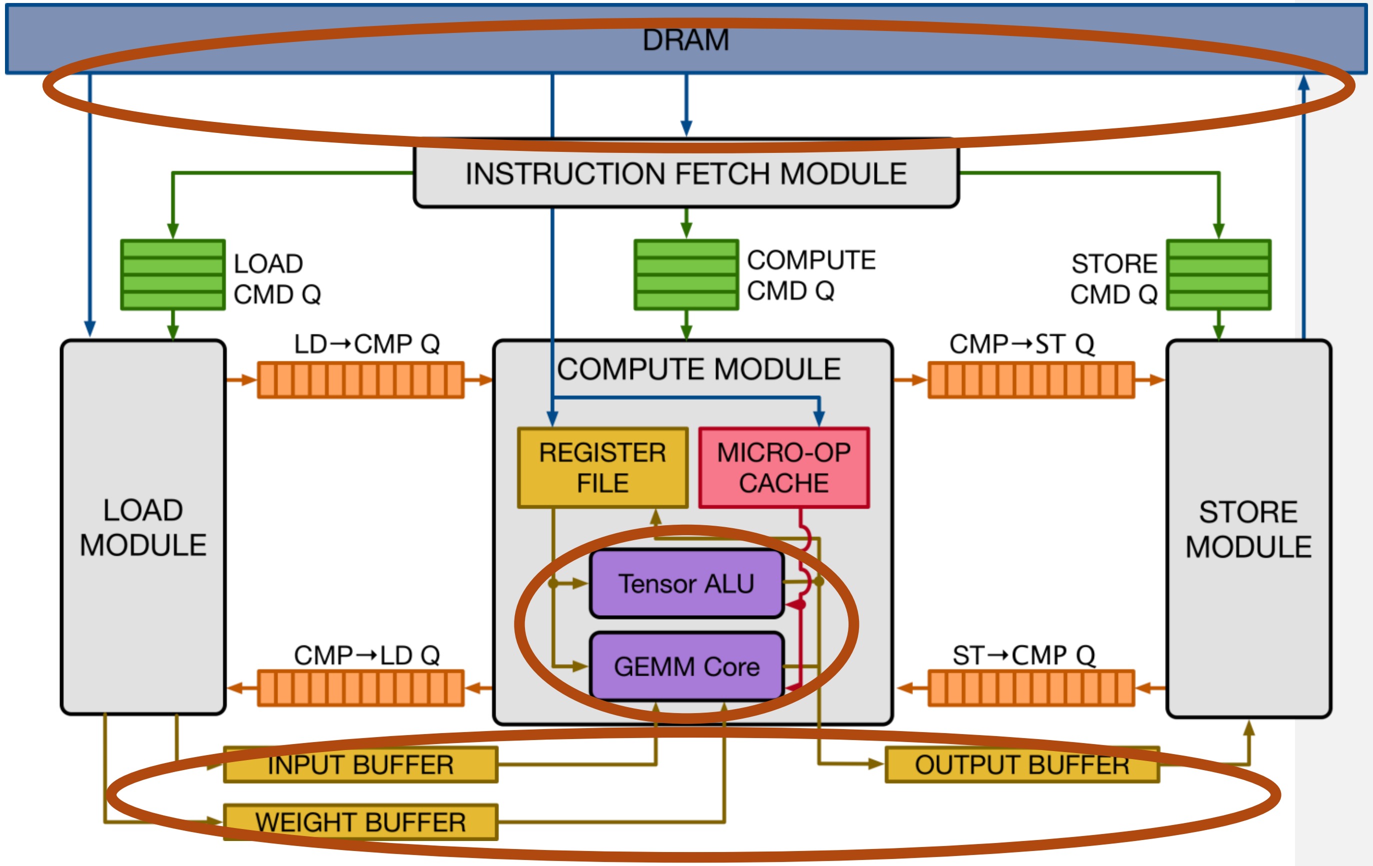
Goals:
Increase size of the VTA design space and enable higher performance

Outline

- Background: VTA
- Increasing Throughput
- Expanded Design Space
- Results
- Conclusion

Background: Versatile Tensor Accelerator (VTA) [1]

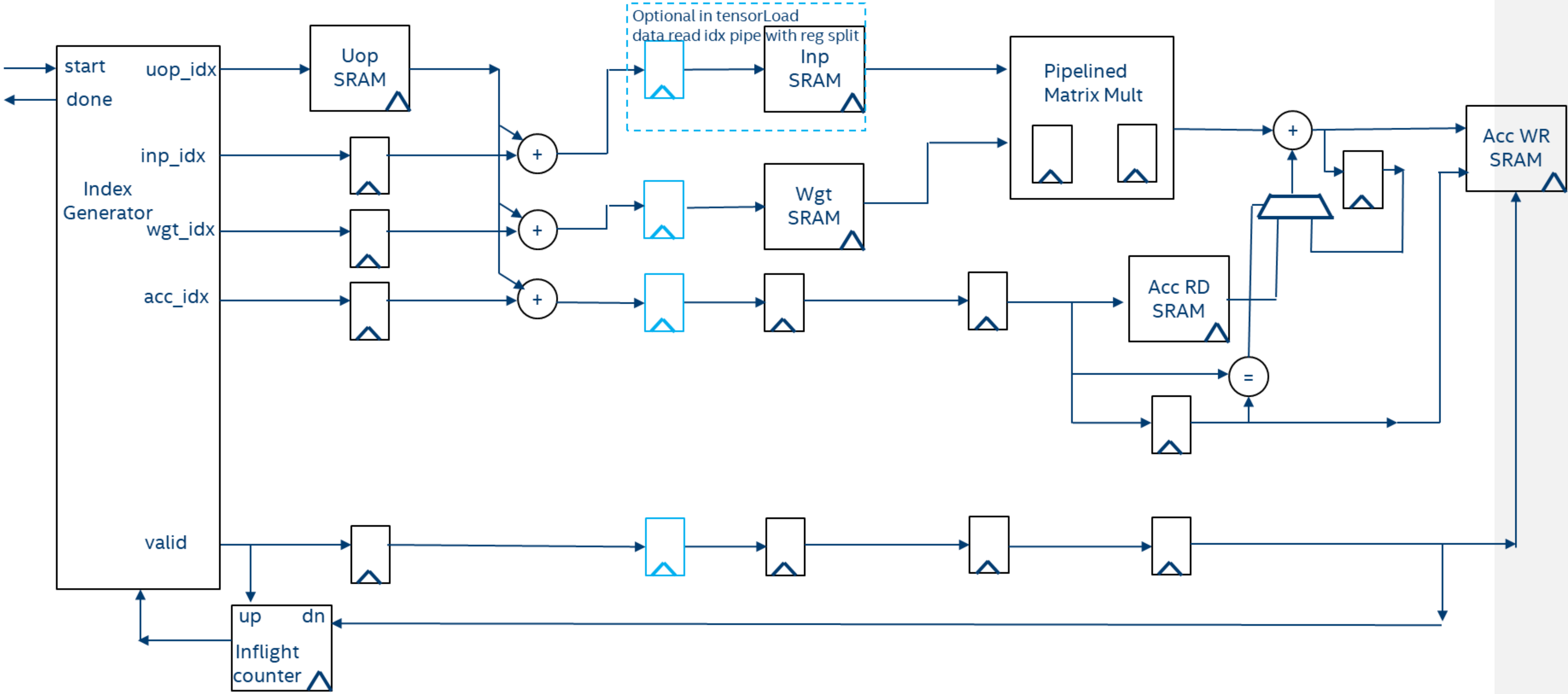
- DNN Inference Accelerator
  - 8-bit input/weight, 32-bit acc
  - GEMM and ALU units
  - Decoupled-Access-Execute [2] uArch with load/compute/store parallelism
- Multiple targets
  - **fsim:** behavioral, C++
  - **tsim:** cycle-accurate, CHISEL
  - Others: pynq, de10, focl, etc
- CHISEL [3]
  - Hardware construction language which produces RTL



Balance execution unit shape/throughput, DRAM access, scratchpad size

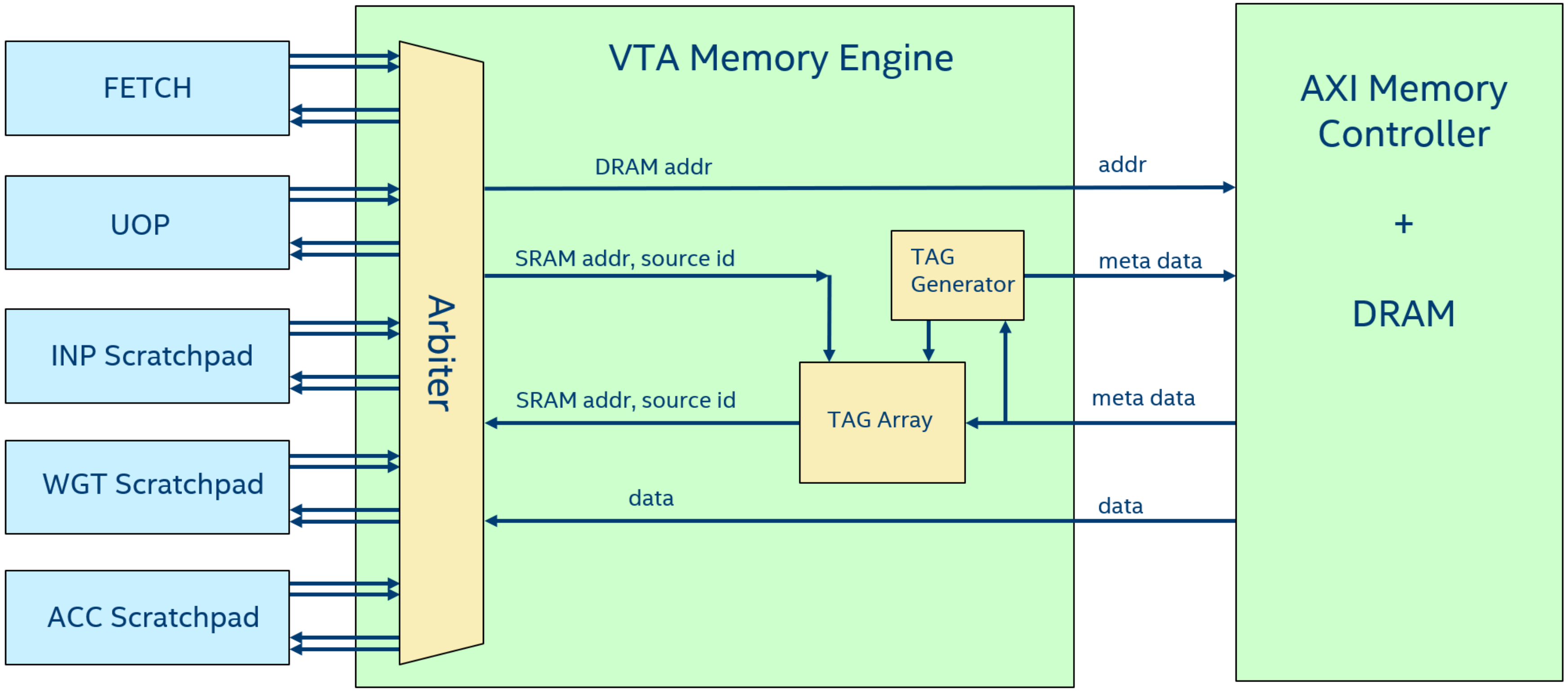
[1] Moreau, Thierry, et al. "A hardware–software blueprint for flexible deep learning specialization." *IEEE Micro* 39.5 (2019): 8-16.
[2] Smith, James E. "Decoupled access/execute computer architectures." ACM Transactions on Computer Systems (TOCS) 2.4 (1984): 289-308.
[3] Bachrach, Jonathan, et al. "Chisel: constructing hardware in a scala embedded language." *DAC Design Automation Conference 2012*. IEEE, 2012.

Pipelining



- GEMM datapath in tsim was pipelined, but control logic was not
- Added pipelined control (index generator) to reduce initiation interval to 1 cycle, resulting in a ~4x throughput increase
- Similar changes for ALU, but acc src/dest port restriction
- Optional flops added to meet frequency targets for larger GEMM shapes

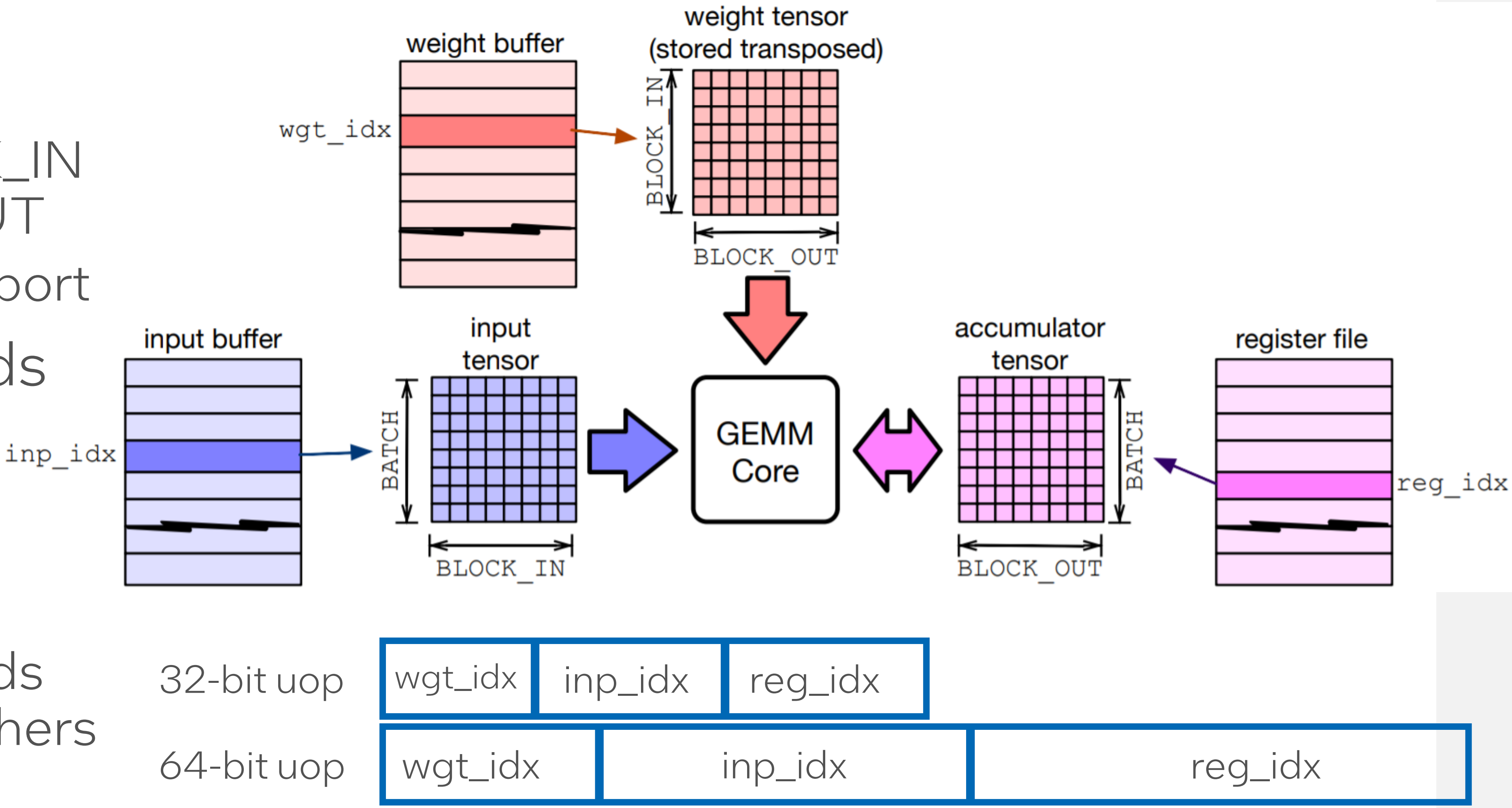
Expanded Design Space: More Capable Memory Interface



- Added configurable memory width (8-64 bytes)
- Now allow multiple outstanding transactions with out-of-order completion

Expanded Design Space: GEMM and scratchpads

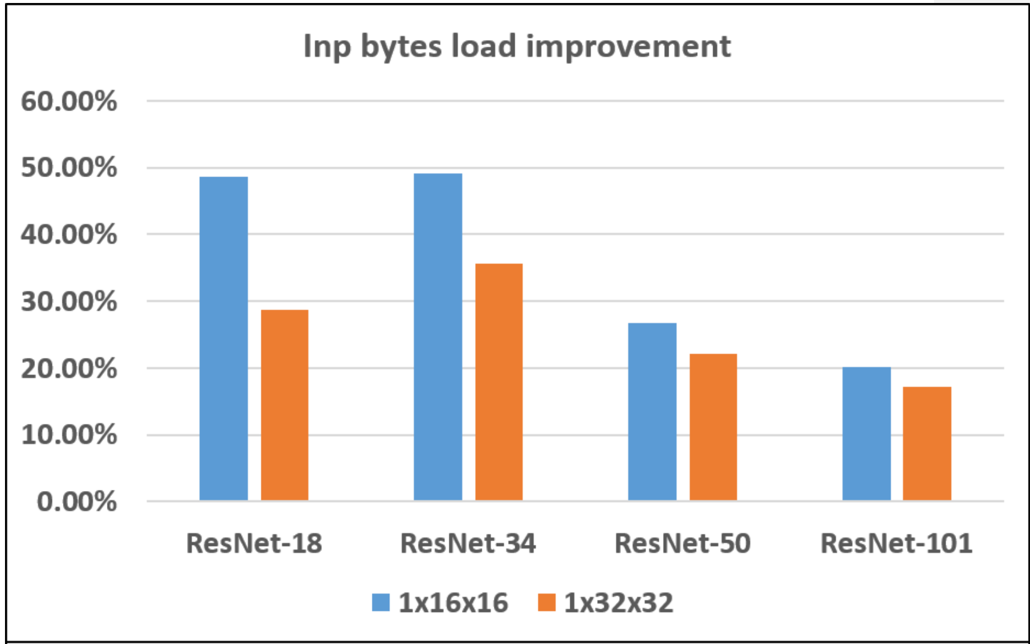
- GEMM shape
  - Separate BLOCK_IN from BLOCK_OUT
  - Add BATCH support
- Larger scratchpads
  - Also permit 64-bit uops
  - Retain 128-bit instruction width: expand some fields while shrinking others

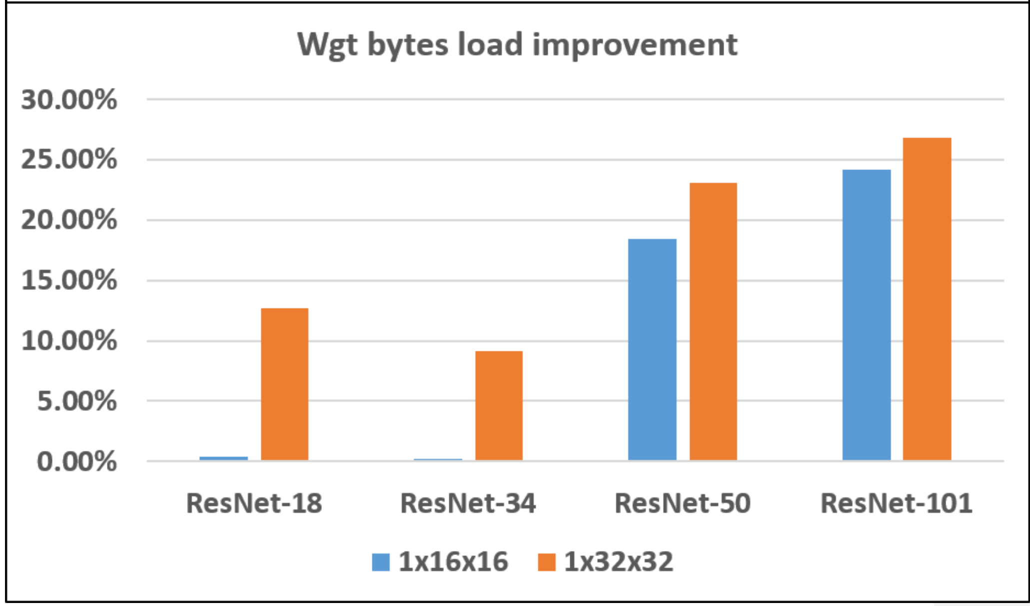


Moreau, Thierry, et al. "VTA: an open hardware-software stack for deep learning." *arXiv preprint arXiv:1807.04188* (2018).

Expanded Design Space: Compiler

- Tiling Parameter Search
  - Heuristic guided analytical scheme for mapping conv2D and depthwise conv2d to VTA configurations
    - Simple memory model allows us to analytically estimate cycle count impact of scheduling decisions
    - With vastly more configurations, measurement is expensive
  - VTA previously used a measurement-guided database of optimal schedules per configuration from AutoTVM
- Double buffering improvements
  - Previous scheme already allowed overlap in load/compute/store execution
  - Now enhanced to allow greater reuse of scratchpad data





Additional Layers

- Depthwise-Conv enables Mobilenet 1.0 [1]
  - Elementwise 8-bit multiplication instruction added
- Pooling support allows FC layer in ResNets [2] to be offloaded to VTA as well
  - Max Pooling
    - Load instruction augmented with parameterized pad values: 0 and MAX_NEG
  - Average Pooling
    - Approximate division using ALU shifts and adds

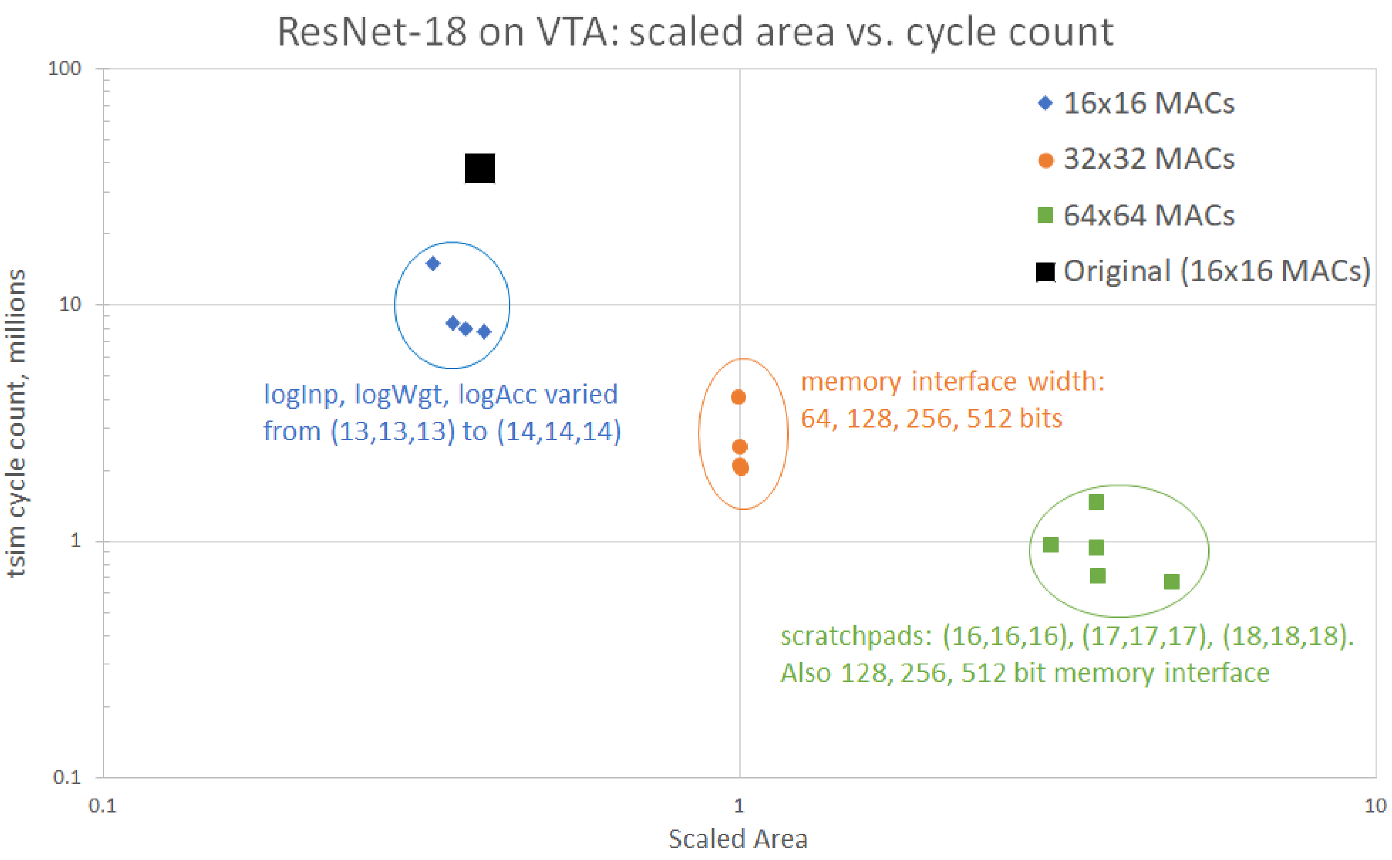
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

layer name	output size	18-layer	34-layer	50-layer	101-layer
conv1	112×112	7×7, 64, stride 2			
conv2_x	56×56	3×3 max pool, stride 2			
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax			
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9

[1] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
[2] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

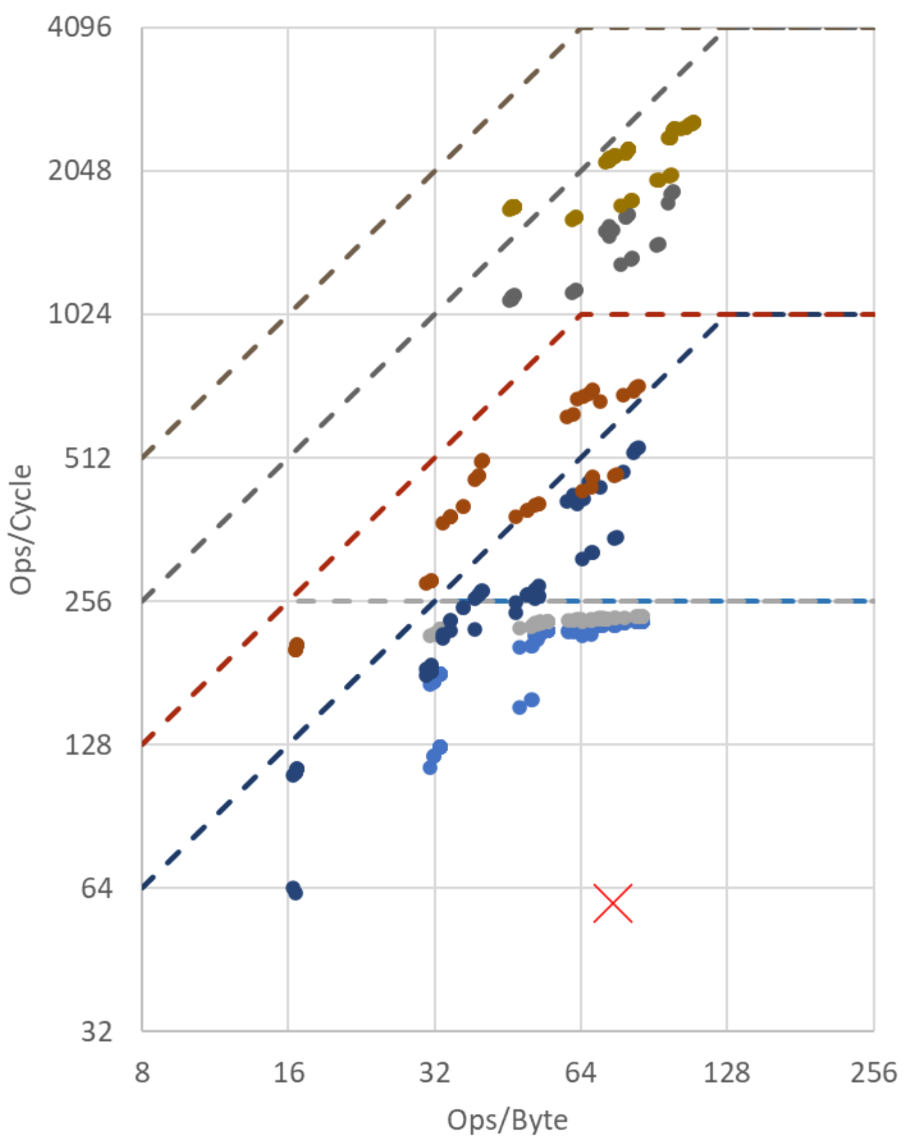
Results: Cycle count vs. Scaled area



- Scaled area is for an ASIC process
  - Using split register files for larger configurations
- GEMM/ALU pipelining
  - ~4x reduction in cycle count
  - Minimal area change
- Must balance compute vs. scratchpad/bandwidth
- Not shown here:
  - Batch size > 1
  - Rectangular GEMM shapes

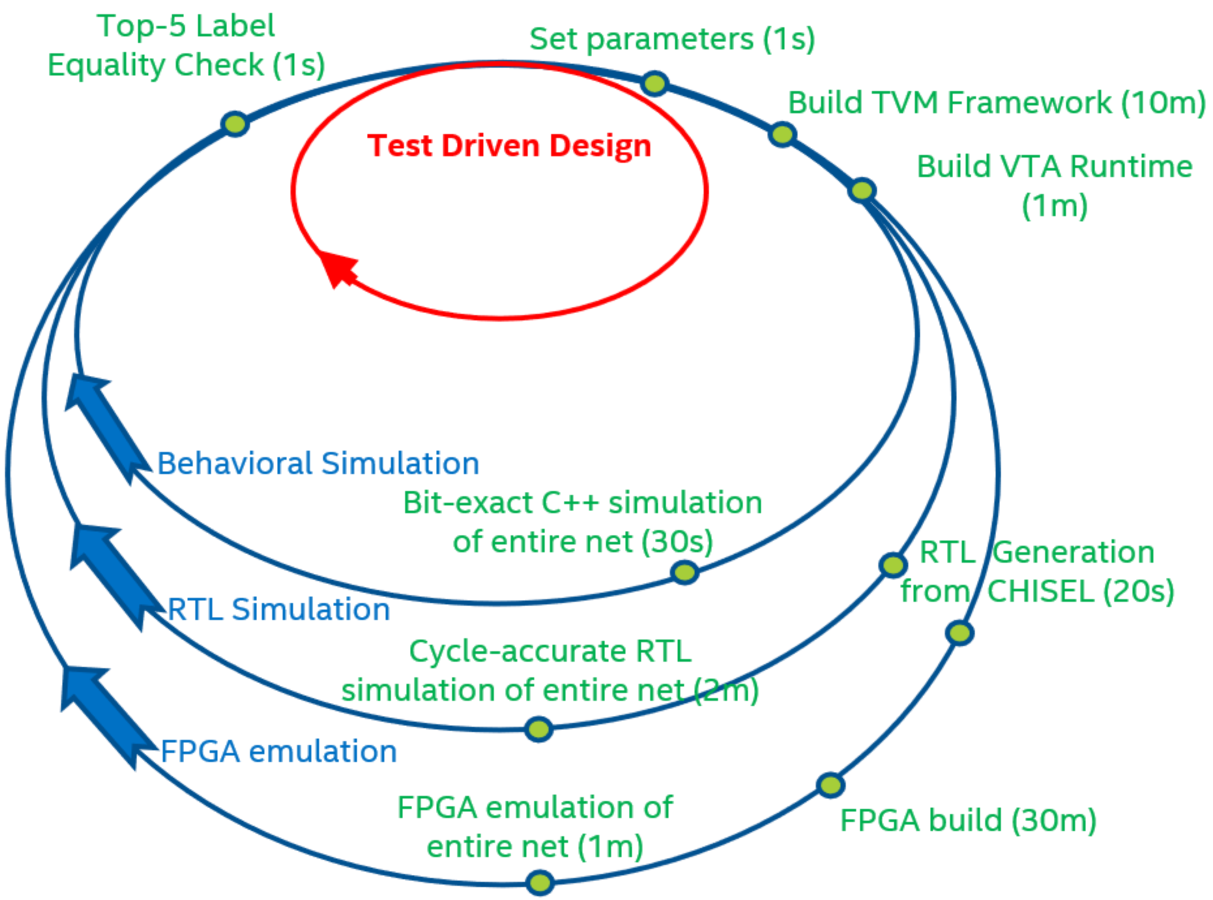
Results: Roofline Analysis

- Roofline plot [1] identifies compute and communication bottlenecks
  - Compute: y-axis, GEMM MAC count
  - Communication: diagonal, bytes/cycle of memory bandwidth
- Each point represents a ResNet-18 RTL simulation
- Much closer to theoretical maximums
  - Lower MACs are compute bound,
  - Higher MACs are memory bound



[1] Williams, Samuel, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures." *Communications of the ACM* 52.4 (2009): 65-76.

Results: Continuous Integration



- Test-Driven design (TDD) loop consists of CHISEL unit tests
  - Initially used for development
  - Valuable for hardware coverage
- Runtimes shown for ResNet-18 workload
- Cycle-accurate tsim simulation is surprisingly fast
  - Most VTA features can be simulated in under 3.5 minutes
  - Parallelization not restricted by physical devices or licensing
- Behavioral simulation helps debug
  - Dynamic Trace-Based Validation

Conclusion

- Enhanced performance and expanded design space
- Open source contributions:
  - Pipelined GEMM, pipelined ALU, and memory interface changes have been upstreamed to the tvm-vta repository
  - All other changes are available in a fork
- Read our arXiv paper to learn more:
  - https://arxiv.org/abs/2111.15024
- Our research focus is improving design methodologies
  - We are hiring!
- Contact me with comments/questions at firstname.lastname@intel.com

intel®